Amendments to Specification:

Please replace the three paragraphs beginning at page 3, line 9 and ending at page 4, line 17 with the following amended paragraphs.

To make this vision a reality, the principles of AIN have been discarded in favor of a new service application component development paradigm. Specifically, it has been recognized that future integrated networks must offer application developers a set of standard, open APIs so that applications written for compatibility with one vendor's system can execute in the system of another vendor. In consequence, the cost of applications development can be amortized, reducing the final cost to the customer. JAVA programming language ~~Java~~™ APIs for Integrated Networks (JAIN) fulfills the requirements of the new service application component development paradigm. Presently, JAIN includes standard, open published JAVA programming language ~~Java~~™ APIs for next-generation systems consisting of integrated Internet Protocol (IP) or asynchronous transport mode (ATM) networks, PSTN, and wireless networks. The JAIN APIs include interfaces at the protocol level, for different protocols such as Media Gateway Control Protocol (MGCP), Session Initiation Protocol (SIP), and Transactional Capabilities Application Part (TCAP), as well as protocols residing in the higher layers of the telecommunications protocol stack.

JAIN includes a set of integrated network APIs for the JAVA programming language ~~Java~~™ platform and an environment to build and integrate JAIN components into services or applications that work across PSTN, packet and wireless networks. The JAIN approach integrates wireline, wireless, and packet-based networks by separating service-based logic from network-based logic. Figure 1 illustrates a conventional JAIN implementation. As shown in Figure 1, a conventional JAIN implementation

can include a protocol layer 102 which can include interfaces to IP, wireline and wireless signaling protocols. These protocols can include TCAP, ISUP, INAP, MAP, SIP, MGCP, and H.323. The JAIN implementation also can include a signaling layer 103 which can include interfaces to provide connectivity management and call control. The conventional JAIN implementation also can include an application layer 104 for handling secure network access and other external services. Finally, the conventional JAIN implementation can include a service layer 106 which can include a service creation and carrier grade service logic execution environment (SLEE) 108.

In JAIN, the protocol layer 102 and the signaling layer 103 are based upon a JAVA programming language ~~Java~~™ standardization of specific signaling protocols and provide standardized protocol interfaces in an object model. Additionally, applications and protocol stacks can be interchanged all the while providing a high degree of portability to the applications in the application layer using protocol stacks from different sources. By comparison, the application layer 104 provides a single call model across all supported protocols in the protocol layer 102. Fundamentally, the application layer 104 provides a single state machine for multiparty, multimedia, and multiprotocol sessions for service components in the application layer 104. This state machine is accessible by trusted applications that execute in the application layer 104 through a call control API.

Please replace the paragraph beginning at page 14, line 21 with the following amended paragraph.

In step 360, the service component optionally can post an acknowledgment to the SLEE for routing to the Web server and hypermedia document contained therein. Accordingly, information contained within the acknowledgment can be presented to the user by dynamically creating a Web

page for presenting that information. For example, the Web page can incorporate dynamic web content creation technology such as JAVA programming language ~~Java™~~ Server Pages. In this manner, two-way communications can be established between the SLEE server and the Web server. Notably, the SLEE server can include service components for sending and receiving information to and from servers on the Internet, as well as service components which can contain and change service attributes. As previously mentioned, these service components can communicate with one another via the event handling component of the SLEE server.